AMENDMENTS TO THE CLAIMS

1. (currently amended) A system for providing a JAVA code release infrastructure with granular code patching, comprising:

a computer readable storage medium holding one or more JAVA code patches, each comprising at least one resource unit, each resource unit comprising metadata and file components;

a computer readable storage medium holding one or more to-be-patched JAVA code libraries, each comprising at least one such resource unit;

a computer readable storage medium holding a patch tool, comprising:

a compare module that determines which units in the to-be-patched JAVA code libraries are outdated by comparing the metadata for each such resource unit in the JAVA code patches to the metadata for each such corresponding resource unit in the to-be-patched JAVA code libraries; and

a merge module merging each such resource unit in the JAVA code patches into the to-be-patched JAVA code libraries for each such corresponding resource unit that is out-of-date.

2. (previously presented) A system according to Claim 1, further comprising:

an extract module extracting at least one resource unit from the JAVA code libraries and modifying one or more JAVA archive files that are out-of-date with the at least one extracted resource unit.

3. (previously presented) A system according to Claim 1, further comprising:

a sign module signing the JAVA archive files using a digital certificate.

4. (previously presented) A system according to Claim 1, wherein the one or more JAVA archive files are modified through at least one of creation, revision or deletion.

5. (previously presented) A system according to Claim 1, further comprising:

a source repository storing the source file components;

a staged patch repository storing the one or more JAVA code patches; and

a staged code repository organizing the one or more JAVA code libraries and the JAVA archive files.

6. (previously presented) A system according to Claim 1, further comp,rising:
a resource unit generator processing the file components into at least one such resource unit; and
a packager packaging at least one such resource unit into one or more of the JAVA code patches.

7. (previously presented) A system according to Claim 6, further comprising:
stored JAVA source code provided as the file components.

8. (previously presented) A system according to Claim 7, further comprising:
a compiler compiling at least one JAVA source code file into one or more JAVA classes; and
a resource unit packager module storing the JAVA classes into at least one such resource unit as the file components.

9. (previously presented) A system according to Claim 6, further comprising:
at least one of non-JAVA source and derived code provided as the file components.

10. (original) A system according to Claim 6, further comprising:
third party code provided as the file components.

11. (original) A system according to Claim 6, further comprising:
a metadata generator generating the metadata for each such resource unit; and
a resource unit packager module storing the generated metadata into the resource unit.

12. (original) A system according to Claim 11, wherein the metadata comprises at least one of a unique identifier and a version attribute.

13. (previously presented) A system according to Claim 1, further comprising:
a compare module using a set of rules allowing one of an older resource unit to be replaced by a newer resource unit and a newer resource unit to be replaced by an older resource unit to back out a previously-applied JAVA code patch.

14. (previously presented) A system according to Claim 1, further comprising:
one or more JAVA archive files, each comprising at least one resource unit corresponding to one such resource unit in the JAVA code libraries; and
a patch tool referencing JAVA archive file definitions which each correspond to one or more of the JAVA archive files.

15. (previously presented) A system according to Claim 14, further comprising:
an extract module extracting the resource units from the JAVA code libraries into the JAVA archive files for each such corresponding resource unit that is out-of-date.

16. (previously presented) A system according to Claim 15, further comprising:
an extract module referencing third party JAVA code libraries not maintained as part of the infrastructure.

17. (previously presented) A system according to Claim 1, further comprising:
JAVA code libraries implemented as a portable virtual file system which can be used directly by a JAVA Virtual Machine.

18. (previously presented) A system according to Claim 1, further comprising:
a machine portable infrastructure providing support for JAVA language features by encapsulating JAVA inner classes, nested directory structures, native class names, and native character set.

19. (previously presented) A method for providing a JAVA code release infrastructure with granular code patching, comprising:
providing one or more JAVA code patches, each comprising at least one resource unit, each resource unit comprising metadata and file components;

patching one or more to-be-patched JAVA code libraries, each comprising at least one such resource unit;

comparing the metadata for each such resource unit in the JAVA code patches to the metadata for each such corresponding resource unit in the to-be-patched JAVA code libraries; and

merging each such resource unit in the JAVA code patches into the to-be-patched JAVA code libraries for each such corresponding resource unit that is out-of-date.

20. (previously presented) A method according to Claim 19, further comprising:
extracting at least one resource unit from the JAVA code libraries and modifying one or more JAVA archive files that are out-of-date with the at least one extracted resource unit.

21. (previously presented) A method according to Claim 19, further comprising:
signing the JAVA archive files using a digital certificate.

22. (previously presented A method according to Claim 19, wherein the one or more JAVA archive files are modified through at least one of creating, updating or deleting.

23. (previously presented) A method according to Claim 19, further comprising:
storing source file components into a source repository;
storing one or more JAVA code patches into a staged patch repository; and
organizing one or more JAVA code libraries and the JAVA archive files into a staged code repository.

24. (previously presented) A method according to Claim 19, further comprising:
processing the file components into at least one such resource unit; and
packaging at least one such resource unit into one or more of the JAVA code patches.

25. (previously presented) A method according to Claim 24, further comprising:
providing JAVA source code as the file components.

26. (previously presented) A method according to Claim 25, further comprising:

compiling at least one JAVA source code file into one or more JAVA classes; and
storing the JAVA classes into at least one such resource unit as the file components.

27. (previously presented) A method according to Claim 24, further comprising:
providing at least one of non-JAVA source and derived code as the file components.

28. (original) A method according to Claim 24, further comprising:
providing third party code as the file components.

29. (original) A method according to Claim 24, further comprising:
generating the metadata for each such resource unit; and
storing the generated metadata into the resource unit.

30. (original) A method according to Claim 29, wherein the metadata comprises at least one of a unique identifier and a version attribute.

31. (previously presented) A method according to Claim 19, further comprising:
using a set of rules to allow one of an older resource unit to be replaced by a newer resource unit and a newer resource unit to be replaced by an older 3 resource unit to back out a previously-applied JAVA code patch.

32. (previously presented) A method according to Claim 19, further comprising:
providing one or more JAVA archive files, each comprising at least one resource unit corresponding to one such resource unit in the JAVA code libraries; and
referencing JAVA archive file definitions which each correspond to one or more of the JAVA archive files.

33. (previously presented) A method according to Claim 32, further comprising:
extracting the resource units from the JAVA code libraries for each such corresponding resource unit that is out-of-date.

34. (previously presented) A method according to Claim 33, further comprising:

referencing third party JAVA code libraries not maintained as part of the infrastructure.

35. (previously presented) A method according to Claim 19, further comprising:
implementing JAVA code libraries as a portable virtual file system which can be used directly by a JAVA Virtual Machine.

36. (previously presented) A method according to Claim 19, further comprising:
providing a machine portable infrastructure supporting JAVA language features by encapsulating JAVA inner classes, nested directory structures, native class names, and native character set.

37. (original) A computer-readable storage medium holding code for performing the method of Claim 19.

38. (previously presented) A system for patching staged code in a staged JAVA code release infrastructure, comprising:
a staged code repository maintaining one or more staged to-be-patched JAVA code libraries, each staged to-be-patched JAVA code library comprising at least one resource unit, each resource unit comprising metadata and file components;
a staged patch repository storing one or more JAVA code patches, each JAVA code patch comprising at least one resource unit corresponding to one such resource unit specified in a JAVA code patch definition; and
a patch tool accessing one or more JAVA code patches in the staged patch repository, comprising:
a compare module comparing the metadata for each resource unit in the JAVA code patches to the metadata in the staged to-be-patched JAVA code libraries for each such corresponding resource unit; and
a merge module merging each resource unit in the JAVA code patches into the staged to-be-patched JAVA code libraries for each such corresponding resource unit that is out-of-date.

39. (previously presented) A system according to Claim 38, further comprising:

an extract module referencing JAVA archive file definitions which each correspond to a staged JAVA archive file, each staged JAVA archive file comprising at least one resource unit corresponding to one such resource unit in the staged JAVA code libraries.

40. (previously presented) A system according to Claim 39, further comprising:
an extract module extracting one such resource unit from the staged JAVA code libraries into the staged JAVA archive files for each such corresponding resource unit that is out-of-date.

41. (previously presented) A system according to Claim 40, further comprising:
a sign module creating a digital signature for the staged JAVA archive files using a digital certificate.

42. (previously presented) A method for patching staged code in a JAVA code release infrastructure, comprising:
maintaining one or more staged to-be-patched JAVA code libraries in a staged code repository, each staged to-be-patched JAVA code library comprising at least one resource unit, each resource unit comprising metadata and file components;
accessing one or more JAVA code patches in a staged patch repository, each JAVA code patch comprising at least one resource unit corresponding to one such resource unit specified in a JAVA code patch definition;
comparing the metadata for each resource unit in the JAVA code patches to the metadata in the staged to-be-patched JAVA code libraries for each such corresponding resource unit; and
merging each resource unit in the JAVA code patches into the staged to-be-patched JAVA code libraries for each such corresponding resource unit that is out-of-date.

43. (previously presented) A method according to Claim 42, further comprising:
referencing JAVA archive file definitions which each correspond to a staged JAVA archive file, each staged JAVA archive file comprising at least one resource unit corresponding to one such resource unit in the staged JAVA code libraries.

44. (previously presented) A method according to Claim 43, further comprising:
extracting one such resource unit from the staged JAVA code libraries into the staged JAVA archive files for each such corresponding resource unit that is out-of-date.

45. (previously presented) A method according to Claim 44, further comprising:
creating a digital signature for the staged JAVA archive files using a digital certificate.

46. (original) A computer-readable storage medium holding code for performing the method of Claim 42.

47-63. (canceled)

64. (previously presented) The system of Claim 1, wherein the compare module resides on a client machine that is separate from a server machine on which resides a patch generator that generated the one or more JAVA code patches.

65. (previously presented) The system of Claim 64, wherein the client machine downloads the one or more JAVA code patches from the server machine.

66. (previously presented) The method of Claim 19, wherein the step of comparing is performed on a client machine that is separate from a server machine on which resides a patch generator that generated the one or more JAVA code patches.

67. (currently amended) The method of Claim 66, further comprising downloading the one or more JAVA code patches from the server ~~client~~ machine to the client ~~server~~ machine.

68. (previously presented) The system of Claim 38, wherein the compare module resides on a client machine that is separate from a server machine on which resides a patch generator that generated the JAVA code patches.

69. (previously presented) The method of Claim 42, wherein the step of comparing is performed on a client machine that is separate from a server machine on which resides a patch generator that generated the JAVA code patches.

70. (previously presented) A computer-implemented method for patching code, the method comprising:

downloading, to a client machine, from a server machine that is separate from the client machine, one or more patches that were generated on the server machine;

comparing, at the client machine, (a) contents of the one or more patches to (b) code that is not resident on the server machine;

based on the comparing, determining one or more out-of-date portions within the code; and

applying at least a part of the one or more patches to the out-of-date portions without applying any part of the one or more patches to parts of the code other than the out-of-date portions.